



**HP64000
Logic Development
System**

**Model 64841A
Assembler Supplement
6800/6801/6802/
6803/8861/6301**



CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

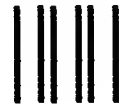
THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

FOLD HERE



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

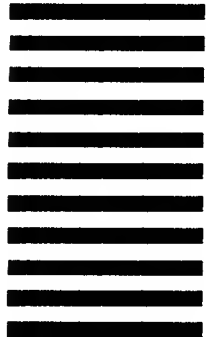
Logic Product Support Dept.

Attn: Technical Publications Manager

Centennial Annex - D2

P.O. Box 617

Colorado Springs, Colorado 80901-0617



FOLD HERE

Your cooperation in completing and returning this form
will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64841A
Assembler Supplement 6800/6801/6802/6803/8861/6301
64841-90905, August 1984

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough
(what more do you need?)

1 2 3 4 5

Covers everything

2. The information in this book is accurate:

Too many errors

1 2 3 4 5

Exactly right

3. The information in this book is easy to find:

I can't find things I need

1 2 3 4 5

I can find info quickly

4. The Index and Table of Contents are useful:

Helpful

1 2 3 4 5

Missing or inadequate

5. What about the "how-to" procedures and examples:

No help

1 2 3 4 5

Very helpful

Too many now

1 2 3 4 5

I'd like more

6. What about the writing style:

Confusing

1 2 3 4 5

Clear

7. What about organization of the book:

Poor order

1 2 3 4 5

Good order

8. What about the size of the book:

too big/small

1 2 3 4 5

Right size

Comments: _____

Particular pages with errors?

Name (optional): _____
Job title: _____
Company: _____
Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

FOLD HERE



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

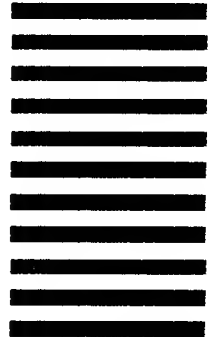
BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

Logic Product Support Dept.
Attn: Technical Publications Manager
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617



FOLD HERE

Your cooperation in completing and returning this form
will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64841A
Assembler Supplement 6800/6801/6802/6803/8861/6301
64841-90905, August 1984

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough
(what more do you need?)

1 2 3 4 5

Covers everything

2. The information in this book is accurate:

Too many errors

1 2 3 4 5

Exactly right

3. The information in this book is easy to find:

I can't find things I need

1 2 3 4 5

I can find info quickly

4. The Index and Table of Contents are useful:

Helpful

1 2 3 4 5

Missing or inadequate

5. What about the "how-to" procedures and examples:

No help

1 2 3 4 5

Very helpful

Too many now

1 2 3 4 5

I'd like more

6. What about the writing style:

Confusing

1 2 3 4 5

Clear

7. What about organization of the book:

Poor order

1 2 3 4 5

Good order

8. What about the size of the book:

too big/small

1 2 3 4 5

Right size

Comments: _____

Particular pages with errors?

Name (optional): _____

Job title: _____

Company: _____

Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.



OPERATOR/REFERENCE MANUAL

ASSEMBLER SUPPLEMENT
6800/6801/6802/6803/8861/6301

© COPYRIGHT HEWLETT-PACKARD COMPANY 1980, 1984
LOGIC SYSTEMS DIVISION
COLORADO SPRINGS, COLORADO, U.S.A.

ALL RIGHTS RESERVED

PRINTING HISTORY

Each new edition of this manual incorporates all material updated since the previous edition. Each new or revised page is indicated by a revision (rev) date. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The date on the back cover changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions.

First Printing	January 1980 (Part Number 64841-90902)
Second Printing	November 1980 (Part Number 64841-90904)
Third Edition	August 1984 (Part Number 64841-90905)

TABLE OF CONTENTS

Chapter 1: GENERAL INFORMATION

INTRODUCTION	1-1
PROGRAMMING CONSIDERATIONS	1-1
Microprocessor Architecture	1-2
Accumulating Registers	1-2
Program Counter Register	1-2
Index Register	1-2
Stack Pointer Register	1-2
Register D	1-2
Condition Code Registers	1-2
MODES OF ADDRESSING	1-3
Addressing Operands	1-3
Dual Operands Operation	1-3
Single Operand Operation	1-3
Inherent Addressing	1-3
Immediate Addressing	1-4
Direct and Extended Addressing	1-4
Relative Addressing	1-4
Indexed Addressing	1-5
CONDITION CODES	1-6
Carry/Borrow (C) Register	1-6
Overflow (V) Register	1-6
Zero (Z) Register	1-6
Negative (N) Register	1-7
Interrupt Mask (I) Register	1-7
Half Carry (H) Register	1-7

Chapter 2: OPERAND RULES AND CONVENTIONS

GENERAL INFORMATION	2-1
ADDITIONAL OPERAND INFORMATION	2-2
Immediate Addressing indicator	2-2
Indexed Addressing Indicator	2-3
Direct-Extended Addressing	
Mode Default	2-3
LOCATION COUNTER INDICATOR	2-4
OPERAND EXPRESSIONS	2-4

TABLE OF CONTENTS (Cont'd)

Chapter 3: SPECIAL PSEUDO INSTRUCTIONS

INTRODUCTION	3-1
BASE__SEG, BASE__END Instruction	3-2
BEXT Instruction	3-3
BSZ Instruction	3-4
DIRECT Instruction	3-4
EXTEND Instruction	3-5
FCB Instruction	3-5
FCC Instruction	3-6
FDB Instruction	3-6
RMB Instruction	3-7
SET Instruction	3-8

Chapter 4: ASSEMBLER OUTPUT LISTING

GENERAL	4-1
INPUT/OUTPUT FILES	4-1
Source Input File	4-1
Assembler Output Files	4-2
OUTPUT LISTING EXAMPLES	4-2

Chapter 5: INSTRUCTION SET SUMMARY

GENERAL	5-1
PREDEFINED SYMBOLS	5-3

Appendix A

8861 MICROPROCESSOR	A-1
---------------------------	-----

Appendix B

6301 MICROPROCESSOR	B-1
---------------------------	-----

LIST OF TABLES

1-1. Instruction Addressing Modes	1-7
4-1. Source Program Format Example	4-3
4-2. Assembler Output Listing	4-4
4-3. Assembler Output with Errors	4-6
5-1. Instruction Set Summary	5-4

Chapter 1

GENERAL INFORMATION (6800/6801/6802/6803/8861/6301)

INTRODUCTION

NOTE

Use the microprocessor number for the assembler directive; i.e., "6800", "6801", "6802", "6803", "8861", or "6301".

This chapter contains general information about the 6800/6801/6802/6803 microprocessors. It briefly discusses the microprocessors architecture, addressing modes, and condition codes. Information on the 8861 and 6301 microprocessors is included in Appendix A, and Appendix B respectively. For the detailed description of a particular microprocessor, refer to the manufacturer's User's Manual.

NOTE

If you are unfamiliar with assembly language or assemblers, read Chapter 6 in the Assembler/Linker Manual. That chapter reviews, briefly, assemblers, assembly language, and the numbering systems.

PROGRAMMING CONSIDERATIONS

NOTE

The 'Programming Considerations' section that follows applies primarily to the 6800 microprocessor. Differences between the 6800, 6801, 6802, and 6803 will be noted; otherwise, the descriptions apply to all four microprocessors.

Microprocessor Architecture

There are six registers available in the microprocessor for control of external memory and peripheral devices that may be associated with the target system. These registers are discussed briefly in the following paragraphs.

Accumulating Registers

The microprocessor has two registers that function primarily as accumulators. They are referred to as register A (ACCA) and register B (ACCB). Each register has its own group of instructions and the mnemonic of the source statement specifies which register is to be used. For example:

ROLA – Rotate content of register A to the left.

ROLB – Rotate content of register B to the left.

CLRA – Clear register A.

CLRB – Clear register B.

Program Counter Register

The 16-bit program counter register may specify 65,536 addresses. The lowest address is 0000H and the highest address is FFFFH.

Index Register

The index register is a special-purpose 16-bit register that allows the microprocessor to move data in two-byte groups into or out of memory. The register is also used for the indexed addressing mode of operation which is explained later in this chapter.

Stack Pointer Register

The stack pointer register is another special-purpose 16-bit register that allows the microprocessor to use a section in random access memory (RAM) as a last in, first out (LIFO) file. This is very valuable when using subroutines or when processing interrupts. The register can be loaded from memory, stored in memory, decremented, and incremented.

Register D

The 6801/6803 microprocessors have the ability to concatenate register A and register B to form a 16-bit register D. When used in a register D configuration, all previous information that was held in register A and register B will be destroyed.

Condition Code Registers

The microprocessor has six condition codes that make up bits 0 through 5 of an 8-bit register. Bits 6 and 7 are not used and are always set to 1. The six condition codes and their use are discussed later in this chapter.

MODES OF ADDRESSING

Addressing Operands

Instructions for a microprocessor may be divided into a number of categories, but their most common attribute is their modes of addressing. An addressing mode refers to the method by which an instruction addresses its operand.

Dual Operands Operation

There are 15 instructions that require two operands for addressing purposes. These instructions are indicated in table 1-1 under the column labeled 'Dual Operand'. The first operand (which is part of the opcode mnemonic) is always an A or a B, designating the required register. The second operand addresses the memory location that is associated with the operation. The format of the second operand must be in accordance with the rules governing immediate, direct, indexed, or extended addressing modes.

Single Operand Operation

There are 3 instructions that require only one operand for addressing either register A or B. These instructions are indicated in table 1-1 under the column labeled 'ACCX'.

For PSH and PUL instructions, the single operand mode of operation (sometimes referred to as the register mode of addressing) is the only valid mode of addressing.

The remaining single operand instructions may be used with indexed or extended addressing.

Inherent Addressing

In a number of instances, the mnemonic instruction specifies one or more registers that contain operands, or where results are to be stored. For example, the ABA opcode requires the two operands that are contained in register A and register B. The opcode also specifies that the result of the operation will be stored in register A.

For some instructions, all information required for addressing is contained in the opcode and no operand field is required. There are 31 such instructions and they are indicated in table 1-1 under the column labeled "Inherent".

Immediate Addressing

In this mode of addressing, the operand of the instruction contains the value to be used in the operation or computation. The only instructions permitted for this mode of addressing are indicated in table 1-1 under the column labeled "Immediate".

To select this mode of addressing, the corresponding operand must be immediately preceded by the pound (#) character. The operand data may be in the form of an ASCII character, a number, a label, or an expression. The data selected for the operand has the following limitations:

#Number, #Symbol, #Expression, #"ASCII Character": For any immediate addressing mode instruction (except CPX, LDS, or LDX), the numeric value must be an integer from 0 to 255. For opcodes CPX, LDS, or LDX, the value range is from 0 to 65,535.

Direct and Extended Addressing

In direct addressing, an instruction requires two bytes of memory. The first byte is the opcode of the instruction and the second byte is the absolute numerical address where the operand is located. Direct addressing allows the user to address memory locations 0 through 255.

In extended addressing, the instruction uses three bytes of memory with the first byte containing the opcode of the instruction, the second byte containing the highest 8 bits of the absolute numerical address, and the third byte containing the lowest 8 bits of the absolute numerical address.

For those instructions that use the direct mode of addressing as well as the extended mode of addressing, default is to extended for externals, relocatables, and forward references. The direct mode is used when addresses are in the 0 to FFH range. The default function can be overridden by using the DIRECT pseudo instruction. Once the direct pseudo is inserted in a source program, the direct mode of addressing will be in effect until canceled by an EXTEND pseudo. Refer to Chapter 2 for examples using the direct and extend pseudos.

Relative Addressing

Branch instructions are somewhat different from other instructions in that their associated addresses do not indicate the location of data. Instead, the address indicates the location of the next instruction that is to be executed. This location is acquired by adding the operand of the instruction and the lowest 8 bits of the program counter plus 2.

For the relative addressing mode to be valid, the distance of the branch must fall in the value range of -125 to +129. This relationship between the relative address and the absolute address of the destination of the branch may be expressed by:

$$DA = (PC + 2) + R$$

where:

DA = address of the destination of the branch instruction.

PC = content of the lowest 8 bits of the program counter.

R = the 8-bit, two's complement, binary number stored in the second byte of the instruction.

When it becomes necessary to branch beyond the valid range of a branch instruction use the JMP (unconditional jump) or JSR (jump to subroutine) instruction. These instructions do not use the relative mode of addressing. See table 1-1 for a list of those instructions that use the relative mode of addressing only.

Indexed Addressing

The microprocessor has a 16-bit index register and there are several instructions associated with this register. These instructions store the register content in memory or permit loading the register from memory. In addition, the index register may be incremented or decremented. Its content may also be compared with two consecutive bytes of memory. With these capabilities, the index register makes an excellent address pointer.

Every instruction that involves an operand in memory may use the indexed addressing mode. These instructions are indicated in table 1-1.

When using indexed addressing, the operand of an instruction is determined by the offset address and the number in the index register. Specifically, the 8-bit offset address is added to the lowest 8 bits in the index register. The sum of this addition becomes the address of the operand.

If a symbol or expression is used, rather than a number, the assembler will compute a numerical value for the symbol or expression. Only values from 0H to 0FFH are valid.

CONDITION CODES

The condition-code register contains codes that are relevant to the execution of instructions. The register is actually a group of one-bit registers that contain the following information:

BIT NO.	CONDITION	
	CODE	DEFINITION
0	C	carry-borrow
1	V	overflow
2	Z	zero
3	N	negative
4	I	interrupt mask
5	H	half-carry

The effect of each instruction on the condition codes is indicated in table 5-1, Chapter 5. A brief description of each condition code is given in the following paragraphs.

Carry/Borrow (C) Register

The carry-borrow register operates like an extension of the A or B register. In an arithmetic addition operation, the final sum may be 9 bits. If this occurs, the carry-borrow code is set (C=1) to indicate a carry. If there was no carry, the C register will be reset (C=0). For the arithmetic subtraction operation, the carry-borrow code represents a borrow condition. The condition code, when set (C=1), indicates that a borrow condition occurred; when reset (C=0), it indicates that there was no borrow.

Overflow (V) Register

The overflow condition code register will be set (V=1) when a two's complement overflow occurs from an arithmetic operation. If no overflow occurs, the register will be reset (V=0).

Zero (Z) Register

The zero register monitors the particular register (A or B) involved in a specific operation. Immediately after the operation, the zero-detect circuit will look at the resulting number. If all zeros are detected, the zero register will be set (Z=1); otherwise, the zero register will be reset (Z=0).

Negative (N) Register

Negative numbers are expressed in the two's complement form with bit 7 indicating the negative quality. Bit 7 will be a 1 if the two's complement was negative. Immediately after an operation that involves the register (A or B), the negative register will look at bit 7 to determine if the result was negative. If so, the condition code (N) will be set (N=1). The condition code will be reset (N=0) if bit 7 was zero, indicating that the two's complement number represented by the result was zero or positive.

Interrupt Mask (I) Register

The interrupt mask code is set (I=1) to prevent the microprocessor from servicing additional external interrupt requests. Interrupt requests from any peripheral device will be ignored by the microprocessor until the interrupt mask code is reset (I=0).

Half Carry (H) Register

The half carry code will be set (H=1) during execution of an ABA, ADC, or ADD instruction if there was a carry from bit position 3 to bit position 4. The half carry code will be reset (H=0) during these instructions if there was no carry from bit position 3.

Table 1-1. Instruction Addressing Modes

D										D									
I U										I U									
N A										N A									
S L I										S L I									
T M E I R										T M E I R									
R O M X I N E										R O M X I N E									
U P E D T N H L										U P E D T N H L									
C E D I E D E A										C E D I E D E A									
T R A I R N E R T										T R A I R N E R T									
I A C A E D X E I										I A C A E D X E I									
O N C T C E E N V										O N C T C E E N V									
N D X E T D D T E										N D X E T D D T E									
ABA										INS									
*ABX										INX									
ADC X X X X X										JMP X X									
ADD X X X X X										JSR X X X									
*ADDD X X X X X										LDA X X X X X									
AND X X X X X										*LDD X X X X X									
ASL X X X										LDS X X X X									

Table 1-1. Instruction Addressing Modes (Cont'd)

D										D									
I U										I U									
N A										N A									
S L I										S L I									
T M E I R										T M E I R									
R O M X I N E										R O M X I N E									
U P E D T N H L										U P E D T N H L									
C E D I E D E A										C E D I E D E A									
T R A I R N E R T										T R A I R N E R T									
I A C A E D X E I										I A C A E D X E I									
O N C T C E E N V										O N C T C E E N V									
N D X E T D D T E										N D X E T D D T E									
*ASLD								X		LDX		X	X	X	X				
ASR	X			X	X					LSR	X			X	X				
BCC								X		*LSRD									X
BCS								X		*MUL									X
BEA								X		NEG	X			X	X				
BGE								X		NOP									X
BGT								X		ORA	X		X	X	X	X			
BHI								X		PSH	X								
*BHS								X		*PSHX									X
BIT	X		X	X	X	X				PUL	X								
BLE								X		*PULX									X
*BLO								X		ROL	X			X	X				
BLS								X		ROR	X			X	X				
BLT								X		RTI									X
BMI								X		RTS									X
BNE								X		SBA									X
BPL								X		SBC	X		X	X	X	X			
BRA								X		SEC									X
*BRN								X		SEI									X
BSR								X		SEV									X
BVC								X		STA	X			X	X	X			
BVS								X		*STD	X			X	X	X			
CBA								X		STS				X	X	X			
CLC								X		STX				X	X	X			
CLI								X		SUB	X			X	X	X			
CLR	X				X	X				*SUBD	X			X	X	X			
CLV								X		SWI									X
CMP	X		X	X	X	X				TAB									X
COM		X			X	X				TAP									X
CPX			X	X	X	X				TBA									X

Table 1-1. Instruction Addressing Modes (Cont'd)

[illegible]

(*) Indicates instruction is only applicable to the 6801 and the 6803 microprocessors.

Chapter2

OPERAND RULES AND CONVENTIONS

GENERAL INFORMATION

The type of information that is placed in the operand field depends on the mnemonic instruction. There are four types of data that may be used in the operand field:

- a. **Register Information** - operands may reference directly data contained in the processor registers such as the stack, register A and B, or the index register.

Example:

```
STAA      SAM      ;MOVE CONTENTS OF  
                ;REGISTER A TO SAM
```

- b. **Index Register Information** - operands may reference directly data contained in the index register.

Example:

```
LDX      0100H      ;LOAD INDEX REGISTER  
                ;FROM MEMORY AS FOLLOWS:  
  
                ;IXH  <--M 0100H DATA  
  
                ;IXL  <--M 0101H DATA
```

- c. **Immediate Data** - operands may contain immediate data. The required value is inserted directly into the operand field. The value may be in the form of numbers, an expression to be evaluated at assembly time, a symbol, or an ASCII constant enclosed in quotation marks.

Examples:

```
LDAA      #0FFH      ;LOAD "FF" HEX INTO
                        ;REGISTER A

LDAB      #'A'        ;LOAD VALUE OF ASCII
                        ;CONSTANT A (0100000)
                        ;INTO REGISTER B
```

- d. **16-bit Memory Address** - operands may reference a 16-bit absolute memory address within the range of 0 to 65,535 that contains the operand data.

Example:

```
LDX      5FFFH
```

ADDITIONAL OPERAND INFORMATION

Immediate Addressing Indicator

To select the immediate addressing mode, the corresponding operand must be preceded by the pound (#) character. The data following the (#) sign will be assigned one or two bytes of memory, depending on the instruction.

Indexed Addressing Indicator

In this addressing mode, the numerical address will be variable and dependent on the content of the index register. The address will be obtained during program execution, rather than being predetermined by the assembler. The operand field of the source statement contains a numerical value which, when added to the content of the index register, furnishes the numerical address. Also, the operand field may contain a symbol or an expression that the assembler replaced with a value that is added to the content of the index register.

The single character 'X' informs the assembler that the indexed addressing mode is to be used. The value of a number, symbol, or expression used for indexed addressing must fall in the range 0H to FFH to be valid. Indexed addressing is called by using the following format:

```
X
,X
number,X
symbol,X
expression,X
```

Examples:

```
ADCA      0AH,X
           or
ADCA      SAM,X
           or
ADCA      X
```

The format 'X', when used alone, indicates that the address of the operand is the same as the content of the index register.

Direct – Extended Addressing Mode Default ((SEC))

For those instructions that can use both direct and extended modes, the assembler defaults to extended for externals, relocatables, and forward references. The direct mode is used when addresses are in the 0 to FFH range. The default function can be overridden by using the DIRECT pseudo instruction. Once the direct pseudo is inserted in a source program, the direct mode of addressing will be in effect until canceled by an EXTEND pseudo instruction.

While in the direct mode of addressing, individual source program statements may be assigned in the extended mode of addressing by appending the letter "E" to the operand.

Example:

```
CLR      0F10H,E
```

While in the extended mode of addressing (EXTEND pseudo in effect), individual source program statements may be assigned the direct mode of addressing by appending the letter "D" to the operand.

Example:

```
ADD      0FH,D
```

Location Counter Indicator

The program counter contains the address of the current instruction or data statement that is being assembled and the dollar symbol (\$) refers to that location.

Example:

```
JUMP     JMP      $+3      ; JUMP TO ADDRESS  
                                ; 3 BYTES BEYOND  
                                ; FIRST BYTE OF THIS  
                                ; INSTRUCTION
```

Operand Expressions

The operand field may contain an expression consisting of one or more terms acted on by the expression operators listed in Chapter 2 of the Assembler/Linker Manual. A term may be either a symbol, a string constant, a numeric constant, or an expression. The assembler reduces the entire expression to a single value.

Terms within expressions may be connected by the following arithmetic operators:

- a. The plus operator (+) produces the arithmetic sum of its operands.
- b. The minus operator (-) produces the arithmetic difference of its operands or the arithmetic negative of its operand when used alone.
- c. The asterisk operator (*) produces the arithmetic product of the operands.
- d. The slant operator (/) produces the quotient of its operands, discarding any remainder.
- e. An instruction enclosed in parentheses is a legal expression in the operand field.

Care should be taken when using the arithmetic operators since their operational results may affect the condition codes in the condition code registers.

Chapter 3

SPECIAL PSEUDO INSTRUCTIONS

INTRODUCTION

This chapter supplements Chapter 3 in the HP Model 64000 Assembler/ Linker Manual. It lists and defines in detail those assembler instructions that are applicable to the 6800/6801/6802/6803 microprocessors only.

BASE__SEG, BASE__END

Declare Symbols Relocatable and on Base Page

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
	BASE__SEG	BASE__END	

The BASE__SEG and BASE__END pseudo instructions alert the assembler for symbols that will be on base page although they are relocatable.

BASE__SEG only affects labels defined by pseudo instructions FCB, FDB, and RMB.

Example:

LABEL	OPERATION	OPERAND	COMMENT
	DATA		
	BASE__SEG		
JULY	FCB	0	;JULY is DATA ;relocatable and ;flagged as base page.
JUNE	RMB	12	
	BASE__END		;Turns off base ;page flag.
	PROG		
	LDAA	JULY	;Generates base page
	LDAA	JUNE	;reference. Linker
			;checks for errors.
			;Labels must be
			;defined before using
			;or they will not be
			;flagged as base page.
	LDAA	AUGUST	;This will not be on
			;base page, since it
			;is defined out of the
			;BASE__SEG range.
AUGUST	FCB	0	

BEXT

Declare Symbols External and on Base Page

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
[Name]	BEXT	operand[,operand,...]	

The BEXT pseudo instruction declares expression as external and on base page. The linker checks for range errors.

Example:

BEXT	SAM	;SAM is external and ;on base page.
EXT	CHARLIE	;Charlie is external
LDA	SAM	;Generates base page ;reference.
LDA	CHARLIE	;Assembler generates ;extended addressing ;unless told to put ;on base page.

BSZ

Block Storage of Zeros

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
[Name]	BSZ	expression	

The BSZ pseudo instruction allocates a block of bytes. Each byte has an initial value of zero. Expression determines the number of bytes allocated.

An error will be generated if Expression has a value of zero or contains symbols that are undefined, external references, or forward references.

Example:

LABEL	OPERATION	OPERAND	COMMENT
	BSZ	10	;Generates 10 bytes ;of zeros.

DIRECT

Direct Addressing Mode

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
	DIRECT		

Some microprocessor instructions can use either the direct or the extended mode of addressing. Unless otherwise instructed, the assembler defaults to extended addressing. To cancel this default condition, insert the **DIRECT** pseudo instruction into the source program.

EXTEND

Extended Addressing Mode

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
	EXTEND		

The **EXTEND** pseudo instruction selects the extended mode of addressing. To cancel the **EXTEND** instruction, insert the **DIRECT** pseudo instruction into the source program.

FCB

Form Constant Byte

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
[Name]	FCB	expression	

The FCB pseudo instruction will store data in consecutive memory locations starting with the current setting of the program counter. The operand field may contain symbols or expressions that evaluate to one byte (8 bits) numbers in the range 0 through 255.

The label name is optional. If the label name is present, it is assigned the starting value of the program counter, and will reference the byte stored by the FCB instruction.

Example:

LABEL	OPERATION	OPERAND	COMMENT
SAM	FCB	CHARLIE+05H	

FCC

Form Constant Character String

SYNTAX:

LABEL	OPERATION	OPERAND	COMMENT
[Name]	FCC	number, string expression	
		or	
[Name]	FCC	string expression	

The FCC pseudo instruction stores ASCII strings into consecutive bytes of memory. Any printable ASCII character can be included in the string. This pseudo has two formats. In the first format, Number is a decimal constant, which specifies the number of characters contained in string expression. If Number exceeds the characters in String Expression, spaces will be inserted to fill the remainder of the string.

In the second format, FCC specifies the string, which can be any printable ASCII character, within quotation marks ("..."), apostrophe marks ('...'), or carets (^...^).

Example:

LABEL	OPERATION	OPERAND	COMMENT
	FCC	10, "TEXT"	;Generates TEXT in ;ASCII followed by ;6 blanks.
	FCC	"TEXT"	;Only generates TEXT.

FDB

Form Double Byte

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
[Name]	FDB	expression list	

The FDB pseudo instruction will store each 16-bit value from the expression list as an address. The values are stored in memory starting at the current setting of the program counter.

Expressions evaluate to one-word (16 bits) numbers, typically addresses. If an expression evaluates to a single byte, it is assumed to be the low order byte of a 16-bit word where the high order byte is all zeros.

If the label name is present, it is assigned the starting address of the program counter, and thus will reference the first byte stored by the FDB instruction.

Example:

LABEL	OPERATION	OPERAND	COMMENT
SAM	FDB	0B123H	

RMB

Reserve Memory Byte

SYNTAX:

LABEL	OPERATION	OPERAND	COMMENT
[Name]	RMB	expression list	

The RMB pseudo instruction may be used to define a block of memory space. The value of the expression in the operand field specifies the number of bytes to be reserved.

Any symbol appearing in the operand field must be predefined. If the value of the operand expression is zero, no memory is reserved; however, if the optional label name is present, it will be assigned the current value of the program counter.

The RMB instruction reserves space in memory by incrementing the program counter by the value in the operand expression.

Example:

LABEL	OPERATION	OPERAND	COMMENT
[Name]	RMB	expression list	

The RMB pseudo instruction may be used to define a block of memory space. The value of the expression in the operand field specifies the number of bytes to be reserved.

Any symbol appearing in the operand field must be predefined. If the value of the operand expression is zero, no memory is reserved; however, if the optional label name is present, it will be assigned the current value of the program counter.

The RMB instruction reserves space in memory by incrementing the program counter by the value in the operand expression.

Example:

LABEL	OPERATION	OPERAND	COMMENT
SAM	RMB	15	;RESERVE 15 ;BYTES FOR SAM ;ROUTINE

SET

Set Symbol To A Value

SYNTAX

LABEL	OPERATION	OPERAND	COMMENT
Name	SET	expression	

The SET pseudo instruction assigns the value of Expression to Name. The value of Name can be changed later in the program with another SET instruction.

Example:

LABEL	OPERATION	OPERAND	COMMENT
SAM	SET	15	;SAM has a value of ;15.

Chapter 4

ASSEMBLER OUTPUT LISTING

GENERAL

The assembler processes source program modules and produces an output that consists of a source program listing, a relocatable object file, and a symbol cross-reference list. Errors detected by the assembler will be noted in the output listing as error messages. Refer to Appendix D in the Assembler/Linker Manual for a listing of all error codes and their definitions.

INPUT/OUTPUT FILES

Source Input File

Input to the assembler is a source file that is created through the editor. It consists of the following:

EXAMPLE	DESCRIPTION
"6800"	- Assembler directive
Source Code	- Source statements consisting of:
.	
.	Assembler Pseudos - refer to Chapter 3 (Assembler/ Linker Manual)
.	
.	Assembler Instructions - refer to Chapter 5, this Supplement
.	

Assembler Output Files

The assembler produces relocatable object modules that are stored under the same name as the source file but in a format that can be processed by the linker. If an object file does not exist at assembly time, the assembler will create one. If an object file does exist, the assembler will replace it.

List File. The list file is a formatted file that is output to a line printer. It can also be stored in a file or applied to the system CRT display. The list may include:

- a. Source statements with object code.
- b. Error messages.
- c. Summary of errors with a description list.
- d. Symbol cross-reference list.

Symbol Cross-Reference List. All symbols are cross-referenced except local macro labels and parameters. A cross-reference listing contains:

- a. Alphabetical list of program symbols.
- b. Line numbers where symbols are defined.
- c. All references (by line numbers) to symbols in the program.

OUTPUT LISTING EXAMPLES

An example of an assembler output listing is given in table 4-2, using the source program example listed in table 4-1. To illustrate an assembler output listing that contains error messages refer to table 4-3.

NOTE

The source program example was not written as a specific program. It merely lists a group of mnemonics to present a formatted example.

Table 4-1. Source Program Format Example

```

                                "6800" LIST XREF

                                EXT          DSPL6,KYBD6
                                ORG          0B00H
EXEC6                           LDS          #0C00H
                                LDAA         #03H
LP1                             STAA        08H
                                DES
                                BGT         LP1
                                LDAB        #06H
LP2                             JSR        KYBD6
                                BCS         LIGHT
                                TPA
                                PSHA
                                DECB
                                BPL         LP2
                                LDAB        #-01H
G0                              LDX         08H,X
                                STAA        #03H
                                DECB
                                DEX
                                BGT         G0
                                PULA
                                STAA        #08H
                                DECB
LIGHT                          JSR        DSPL6
                                BRA         LP1
                                END
  
```

Table 4-2. Assembler Output Listing

FILE: PGM68E:

HEWLETT PACKARD: MOTOROLA 6800 ASSEMBLER

LINE	LOC	CODE	ADDR	SOURCE STATEMENT
1				"6800" LIST XREF
2				EXT DSPL6,KYBD6
3	0B00			ORG 0B00H
4	0B00	8E	0C00	EXEC6 LDS #0C00H
5	0B03	86	03	LDAA #03H
6	0B05	97	08	LP1 STAA 08H
7	0B07	34		DES
8	0B08	2E	FB	BGT LP1
9	0B0A	C6	06	LDAB #06H
10	0B0C	BD	0000	LP2 JSR KYBD6
11	0B0F	25	13	BCS LIGHT
12	0B11	07		TPA
13	0B12	36		PSHA
14	0B13	5A		DECB
15	0B14	2A	F6	BPL LP2
16	0B16	C6	FF	LDAB #-01H
17	0B18	EE	08	GO LDX 08H,X
18	0B1A	87	03	STAA #03H
19	0B1C	5A		DECB
20	0B1D	09		DEX
21	0B1E	2E	F8	BGT GO
22	0B20	32		PULA
23	0B21	87	03	STAA #03H
24	0B23	5A		DECB
25	0B24	BD	0000	LIGHT JSR DSPL6
26	0B27	20	DC	BRA LP1
27				END

Errors = 0

Table 4-2. Assembler Output Listing (Cont'd)

FILE: PGM68E:

CROSS REFERENCE TABLE

PAGE 2

LINE#	SYMBOL	TYPE	REFERENCES
2	DSPL6	E	25
4	EXEC6	A	
17	GO	A	
2	KYBD6	E	10
25	LIGHT	A	
6	LP1	A	
10	LP2	A	
***	X	U	17

NOTE: In the cross-reference table, the letter listed under the TYPE column has the following definition:

A = Absolute
C = Common (COMN)
D = Data (DATA)
E = External
M = Multiple Defined
P = Program (PROG)
R = Predefined Register
U = Undefined

Table 4-3. Assembler Output with Errors

FILE: PGM68E: HEWLETT-PACKARD: MOTOROLA 6800 ASSEMBLER

LINE	LOC	CODE	ADDR		SOURCE STATEMENT
1					
2					EXT DSPL6,KYBD6
3	0B00				ORG 0B00H
4	0B00	8E	0C00	EXEC6	LDS #0C00H
5	0B03	86	03		LDAA #0803H
ERROR-LR ^					
6	0B05	97	08	LP1	STAA 08H
7	0B07	34			DES
8	0B08	2E	FB		BBB LP1
ERROR-UO, see line 5 ^					
9	0B0A	C6	06		LDAB #06H
10	0B0C	BD	0000	LP2	JSR KYBD6
11	0B0F	25	13		BCS LIGHT
12	0B11	07			TPA
13	0B12	36			PSHA
14	0B13	5A			DEC C
ERROR-US, see line 8 ^					
15	0B14	2A	F6		BPL LP2
16	0B16	C6	FF		LDAB ##-01H
17	0B18	EE	08	GO	LDX 08H,X
18	0B1A	87	03		STAA #03H
19	0B1C	5A			DECB
20	0B1D	09			DEX
21	0B1E	2E	F8		BGT GO
22	0B20	32			PULA
23	0B21	87	03		STAA #08H
24	0B23	5A			DECP
25	0B24	BD	0000	LIGHT	JSR DSPL6
26	0B27	20	DC		BRA LP1
27					END

Errors = 3, previous error at line 14

US - Undefined Symbol, The indicated symbol is not defined
 LR - Legal Range, Address, or displacement is out of range
 UO - Unidentified Opcode, Opcode encountered is not defined

Table 4-3. Assembler Output with Errors (Cont'd)

FILE: PGM68E:

CROSS REFERENCE TABLE

PAGE 2

LINE#	SYMBOL	TYPE	REFERENCES
2	DSPL6	E	25
4	EXEC6	A	
17	GO	A	
2	KYBD6	E	10
25	LIGHT	A	
6	LP1	A	
10	LP2	A	
***	X	U	17

NOTE: Error messages are inserted immediately following the statement where the error occurs. All error messages (after the first error message) will contain a statement that points to the line number where the previous error occurred. At the end of the source program listing, an error summary statement will be printed. The summary will contain a statement indicating the total number of errors noted, along with a line reference to the previous error. It will also define all error codes listed in the source program listing.

The primary purpose of the error statement that points to the line number where the previous error occurred is to facilitate location of errors. Since some programs may be many pages in length, this feature helps the programmer locate errors quickly (as opposed to thumbing through each page of the program).

Chapter 5

INSTRUCTION SET SUMMARY (6800/6801/6802/6803)

GENERAL

All mnemonic instructions are summarized in table 5-1. The table is arranged in alphabetical order.

Each instruction consists of a mnemonic symbol, an object code for each addressing mode, the boolean operation performed, and condition codes affected. The descriptive symbols used in table 5-1 to represent items in the mnemonic definitions are as follows:

SYMBOL	DESCRIPTION
A	Register A
B	Register B
CC	Carry condition flag
CCR	Condition Code Register
D	Register D (Reg A and Reg B)
Dir	Direct addressing mode
Ext	Extended addressing mode
I	Interrupt mask register
Imm	Immediate addressing mode
Ind	Indexed addressing mode

Assembler Supplement 6800/6801/6802/6803/8861/6301
Instruction Set Summary

SYMBOL	DESCRIPTION
Inh	Inherent addressing mode
M	A memory location
M+1	The byte of memory location at 0001 plus the address of the memory loction indicated by 'M'
n	Not affected by the operation.
N	Negative condition code
OP	Operation Code (Hexadecimal)
PC	Program Counter
PCH	Program Counter High Byte
PCL	Program Counter Low Byte
Rel	Relative Address
SP	Stack Pointer
u	Condition code unknown
V	Overflow condition code
x	Affected by the operation
X	Index Register - 16 bits
Z	Zero condition code
0	Bit = 0
1	Bit = 1
	Boolean AND
--> or <--	Transfer into
+	Arithmetic addition
-	Arithmetic subtraction

SYMBOL	DEFINITION
*	Multiply
=	Equality
()	Refers to contents of address or register
\oplus	Exclusive OR
\odot	Inclusive OR

PREDEFINED SYMBOLS

The following symbols are reserved. They have special meaning to the assembler and cannot appear as user-defined symbols.

SYMBOL	DEFINITION
A	Register A
B	Register B
D	Register D (Reg A and Reg B)
X	Index Register
SP	Stack Pointer
\$	Program Counter content

Assembler Supplement 6800/6801/6802/6803/8861/6301
Instruction Set Summary

Table 5-1. Instruction Set Summary

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C					
				BITS	5	4	3	2	1 0
ABA	1B	Inh	$A = (A) + (B)$		x	n	x	x	x x
*ABX	3A	Inh	$X = (X) + (B)$		n	n	n	n	n n
ADCA	89	Imm	$A = (A) + (M) + (C)$		x	n	x	x	x x
	99	Dir							
	B9	Ext							
	A9	Ind							
ADCB	C9	Imm	$B = (B) + (M) + (C)$		x	n	x	x	x x
	D9	Dir							
	F9	Ext							
	E9	Ind							
ADDA	8B	Imm	$A = (A) + (M)$		x	n	x	x	x x
	9B	Dir							
	BB	Ext							
	AB	Ind							
ADDB	CB	Imm	$B = (B) + (M)$		x	n	x	x	x x
	DB	Dir							
	FB	Ext							
	EB	Ind							
*ADDD	C3	Imm	$D = (D) + (M)$		n	n	x	x	x x
	D3	Dir							
	F3	Ext							
	E3	Ind							
ANDA	84	Imm	$A = (A) \bullet (M)$		n	n	x	x	0 n
	94	Dir							
	B4	Ext							
	A4	Ind							
ANDB	C4	Imm	$B = (B) \bullet (M)$		n	n	x	x	0 n
	D4	Dir							
	F4	Ext							
	E4	Ind							
ASL	78	Ext	$CC \leftarrow \neg 7 \leftarrow \neg 0 \leftarrow \neg 0$ (M)		n	n	x	x	x x
	68	Ind							

Table 5-1. Instruction Set Summary (Cont'd)

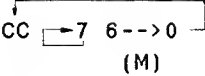
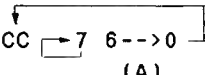
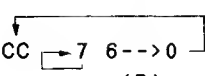
MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS 5 4 3 2 1 0						
ASLA	48	Inh	CC<--7<--0<--0 (A)		n	n	x	x	x	x
ASLB	58	Inh	CC<--7<--0<--0 (B)		n	n	x	x	x	x
*ASLD	05	Inh	CC<--7<--0<--7<--0<--0 (A) (B)		n	n	x	x	x	x
ASR	77	Ext			n	n	x	x	x	x
	67	Ind								
ASRA	47	Inh			n	n	x	x	x	x
ASRB	57	Inh			n	n	x	x	x	x
BCC	24	Rel	Test for CC = 0		n	n	n	n	n	n
BCS	25	Rel	Test for CC = 1		n	n	n	n	n	n
BEQ	27	Rel	Test for Z = 1		n	n	n	n	n	n
BGE	2C	Rel	Test for $N \oplus V = 0$		n	n	n	n	n	n
BGT	2E	Rel	Test for $Z \odot [N \oplus V] = 0$		n	n	n	n	n	n
BHI	22	Rel	Test for $CC \odot Z = 0$		n	n	n	n	n	n
*BHS	24	Rel	Test for CC = 0		n	n	n	n	n	n
BITA	85	Imm	(A) • (M)		n	n	x	x	0	n
	95	Dir								
	B5	Ext								
	A5	Ind								
BITB	C5	Imm	(B) • (M)		n	n	x	x	0	n
	D5	dir								
	F5	Ext								
	E5	Ind								

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS 5 4 3 2 1 0						
BLE	2F	Re1	Test for $Z \odot [N \oplus V] = 1$	n	n	n	n	n	n	n
*BLO	25	Re1	Test for $CC = 1$	n	n	n	n	n	n	n
BLS	23	Re1	Test for $CC \odot Z = 1$	n	n	n	n	n	n	n
BLT	2D	Re1	Test for $N \oplus V = 1$	n	n	n	n	n	n	n
BMI	2B	Re1	Test for $N = 1$	n	n	n	n	n	n	n
BNE	26	Re1	Test for $Z = 0$	n	n	n	n	n	n	n
BPL	2A	Re1	Test for $N = 0$	n	n	n	n	n	n	n
BRA	20	Re1	Branch Always	n	n	n	n	n	n	n
*BRN	21	Re1	Branch Never	n	n	n	n	n	n	n
BSR	8D	Re1	Branch Subroutine	n	n	n	n	n	n	n
BVC	28	Re1	Test for $V = 0$	n	n	n	n	n	n	n
BVS	29	Re1	Test for $V = 1$	n	n	n	n	n	n	n
CBA	11	Inh	$(A) - (B)$	n	n	x	x	x	x	x
CLC	0C	Inh	$CC = 0$	n	n	n	n	n	n	0
CLI	0E	Inh	$I = 0$	n	0	n	n	n	n	n
CLR	7F	Ext	$(M) = 0$	n	n	0	1	0	0	0
	6F	Ind								
CLRA	4F	Inh	$(A) = 0$	n	n	0	1	0	0	0
CLRB	5F	Inh	$(B) = 0$	n	n	0	1	0	0	0
CLV	0A	Inh	$V = 0$	n	n	n	n	0	n	n
CMPA	81	Imm	Compare $(A), (M)$	n	n	x	x	x	x	x
	91	Dir								
	B1	Ext								
	A1	Ind								

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS 5 4 3 2 1 0						
CMPB	C1	Imm	Compare (B), (M)	n n x x x x						
	D1	Dir								
	F1	Ext								
	E1	Ind								
COM	73	Ext	Complement (M)	n n x x 0 1						
	63	Ind								
COMA	43	Inh	Complement (A)	n n x x 0 1						
COMB	53	Inh	Complement (B)	n n x x 0 1						
CPX	8C	Imm	Compare (X), (M)	n n x x x n						
	9C	Dir								
	BC	Ext								
	AC	Ind								
DAA	19	Inh	Converts Reg (A) into packed BCD	n n x x x x						
DEC	7A	Ext	(M) = (M) - 1	n n x x u n						
	6A	Ind								
DECA	4A	Inh	(A) = (A) - 1	n n x x u n						
DECB	5A	Inh	(B) = (B) - 1	n n x x u n						
DES	34	Inh	(SP) = (SP) - 1	n n n n n n						
DEX	09	Inh	(X) = (X) - 1	n n n x n n						
EORA	88	Imm	(A) = (A) \oplus (M)	n n x x 0 n						
	98	Dir								
	B8	Ext								
	A8	Ind								
EORB	C8	Imm	(B) = (B) \oplus (M)	n n x x 0 n						
	D8	Dir								
	F8	Ext								
	E8	Ind								
INC	7C	Ext	(M) = (M) + 1	n n x x u n						
	6C	Ind								

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS	5	4	3	2	1	0
INCA	4C	Inh	(A) = (A) + 1		n	n	x	x	u	n
INCB	5C	Inh	(B) = (B) + 1		n	n	x	x	u	n
INS	31	Inh	(SP) = (SP) + 1		n	n	n	n	n	n
INX	08	Inh	(X) = (X) + 1		n	n	n	x	n	n
JMP	7E	Ext	PC ← -EA		n	n	n	n	n	n
	6E	Ind								
JSR	BD	Ext	(SP) = (SP) - 1,		n	n	n	n	n	n
	AD	Ind	(SP) ← -PCL; (SP) = (SP) - 1, (SP) ← -PCH; PC ← -EA							
LDAA	86	Imm	(A) ← -(M)		n	n	x	x	0	n
	96	Dir								
	B6	Ext								
	A6	Ind								
LDAB	C6	Imm	(B) ← -(M)		n	n	x	x	0	n
	D6	Dir								
	F6	Ext								
	E6	Ind								
*LDD	CC	Imm	(D) ← -(M)		n	n	x	x	0	n
	DC	Dir								
	FC	Ext								
	EC	Ind								
LDS	8E	Imm	(S) ← -(M)		n	n	x	x	0	n
	9E	Dir								
	BE	Ext								
	AE	Ind								
LDX	CE	Imm	(X) ← -(M)		n	n	x	x	0	n
	DE	Dir								
	FE	Ext								
	EE	Ind								
LSR	74	Ext	0 → 7 → 0 → CC		n	n	0	x	x	x
	64	Ind	(M)							

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS 5 4 3 2 1 0						
LSRA	44	Inh	0-->7-->0-->CC (A)	n	n	0	x	x	x	
LSRB	54	Inh	0-->7-->0-->CC (B)	n	n	0	x	x	x	
LSRD	04	Inh	0-->7-->0-->7-->0-->CC (A) (B)	n	n	0	x	x	x	
MUL	3D	Inh	(D)<--(A)(B)	n	n	n	x	n	x	
NEG	70 60	Ext Ind	Two's Complement (M)	n	n	x	x	x	x	
NEGA	40	Inh	Two's Complement (A)	n	n	x	x	x	x	
NEGB	50	Inh	Two's Complement (B)	n	n	x	x	x	x	
NOP	01	Inh	No operation (PC)=(PC)+1	n	n	n	n	n	n	
ORAA	8A 9A BA AA	Imm Dir Ext Ind	(A)<--(A) ⊙ (M)	n	n	x	x	0	n	
ORAB	CA DA FA EA	Imm Dir Ext Ind	(B)<--(B) ⊙ (M)	n	n	x	x	0	n	
PSHA	36	Inh	Push Reg onto stack	n	n	n	n	n	n	
PSHB	37	Inh	Push Reg onto stack	n	n	n	n	n	n	
*PSHX	3C	Inh	Push Index Reg onto stack	n	n	n	n	n	n	
PULA	32	Inh	Pull Reg from stack	n	n	n	n	n	n	

Table 5-1. Instruction Set Summary (Cont'd)

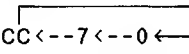
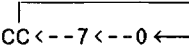
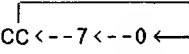
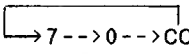
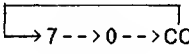
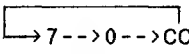
MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C BITS 5 4 3 2 1 0
PULB	33	Inh	Pull Reg from stack	n n n n n n
*PULX	38	Inh	Pull Index Reg from stack	n n n n n n
ROL	79 69	Ext Ind	 (M)	n n x x u x
ROLA	49	Inh	 (A)	n n x x u x
ROLB	59	Inh	 (B)	n n x x u x
ROR	76 66	Ext Ind	 (M)	n n x x u x
RORA	46	Inh	 (A)	n n x x u x
RORB	56	Inh	 (B)	n n x x u x
RTI	3B	Inh	Return from Interrupt	u u u u u u
RTS	39	Inh	Return from Subroutine	n n n n n n
SBA	10	Inh	(A)=(A)-(B)	n n x x x x
SBCA	82 92 B2 A2	Imm Dir Ext Ind	(A)<--(A)-(M)-(CC)	n n x x x x
SBCB	C2 D2 F2 E2	Imm Dir Ext Ind	(B)<--(B)-(M)-(CC)	n n x x x x
SEC	0D	Inh	CC=1	n n r n n 1
SEI	0F	Inh	I=1	n 1 n n n n

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS	5	4	3	2	1	0
SEV	0B	Inh	V=1			n	n	n	n	1 n
STAA	97	Dir	(M) <-- (A)			n	n	x	x	0 n
	B7	Ext								
	A7	Ind								
STAB	D7	Dir	(M) <-- (B)			n	n	x	x	0 n
	F7	Ext								
	E7	Ind								
*STD	DD	Dir	(M) <-- (D)			n	n	x	x	0 n
	FD	Ext								
	ED	Ind								
STS	9F	Dir	(M) <-- (S)			n	n	x	x	0 n
	BF	Ext								
	AF	Ind								
STX	DF	Dir	(M) <-- (X)			n	n	x	x	0 n
	FF	Ext								
	EF	Ind								
SUBA	80	Imm	(A) <-- (A) - (M)			n	n	x	x	x x
	90	Dir								
	B0	Ext								
	A0	Ind								
SUBB	C0	Imm	(B) <-- (B) - (M)			n	n	x	x	x x
	D0	Dir								
	F0	Ext								
	E0	Ind								
*SUBD	83	Imm	(D) <-- (D) - (M)			n	n	x	x	x x
	93	Dir								
	B3	Ext								
	A3	Ind								
SWI	3F	Inh	Software Interrupt			n	n	n	n	n n
TAB	16	Inh	(B) <-- (A)			n	n	x	x	0 n
TAP	06	Inh	(CCR) <-- (A)			x	x	x	x	x x

Assembler Supplement 6800/6801/6802/6803/8861/6301
Instruction Set Summary

Table 5-1. Instruction Set Summary (Cont'd)

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	FLAG H I N Z V C						
				BITS 5 4 3 2 1 0						
TBA	17	Inh	(A) <-- (B)		n	n	x	x	0	n
TPA	07	Inh	(A) <-- (CCR)		n	n	n	n	n	n
TST	7D	Ext	Test (M)-0		n	n	x	x	0	0
	6D	Ind								
TSTA	4D	Inh	Test (A)-0		n	n	x	x	0	0
TSTB	5D	Inh	Test (B)-0		n	n	x	x	0	0
TSX	30	Inh	(X) <-- (SP)+1		n	n	n	n	n	n
TXS	35	Inh	(SP) <-- (X)-1		n	n	n	n	n	n
WAI	3E	Inh	Wait for interrupt		n	x	n	n	n	n

(*) Indicates instruction is applicable to the 6801 and the 6803 microprocessors only.

Appendix A

8861 MICROPROCESSOR

The instructions on the following pages are only applicable to the 8861 microprocessor.

ADX

Add Index Register

SYNTAX

LABEL	OPERATION	OPERAND	OBJECT
			CODE (HEX)
	ADX	#data	0EC
	ADX	address	0FC

Adds the contents of the index register and the contents of memory at the address specified by the program and places the results in the index register. This is a two-byte instruction with immediate addressing (0ECH) or a three-byte instruction with extended addressing (0FC).

Examples:

ADX	#57
ADX	ADDR

NIM

AND Immediate with Memory

SYNTAX

LABEL	OPERATION	OPERAND	OBJECT CODE (HEX)
	NIM	data address,X	71

Performs a logical AND between the contents of the second byte of the instruction and the contents of the memory location specified by the program, and stores the result in the memory location. This is a three-byte instruction using the indexed addressing mode.

Example:

NIM	5FH ADDR,X
-----	------------

OIM

OR Immediate with Memory

SYNTAX

LABEL	OPERATION	OPERAND	OBJECT
			CODE (HEX)
	OIM	data address,X	72

Performs a logical OR between the contents of the second byte of the instruction and the contents of the memory location specified by the program, and stores the result in the memory location. This is a three-byte instruction using the indexed addressing mode.

Example:

OIM 5FH ADDR,X

TMM

Test Under Mask

SYNTAX

LABEL	OPERATION	OPERAND	OBJECT
			CODE (HEX)
	TMM	data address,X	7B

The state of the operand bits selected by a mask is used to set the condition code. The second byte of the instruction is used as an eight-bit mask. The bits of the mask are made to correspond one for one with the bits of the character in memory specified by the index register. A mask bit of one indicates that the memory bit is to be tested. When the mask bit is zero, the memory bit is ignored. This is a three-byte instruction using the indexed addressing mode.

Example:

TMM	5FH ADDR,X
-----	------------

XIM

Exclusive OR Immediate with Memory

SYNTAX

LABEL	OPERATION	OPERAND	OBJECT
			CODE (HEX)
	XIM	data address,X	75

Performs a logical exclusive OR between the contents of the second byte of the instruction and the contents of memory location specified by the program, and stores the result in the memory location. This is a three-byte instruction using the indexed addressing mode.

Example:

XIM	5FH ADDR,X
-----	------------

Appendix B

6301 PROCESSOR

INTRODUCTION

The 6301 processor is UPWARD source and object code compatible with the 6801 processor. The assembler directive is "6301". In addition to the standard 6801 instructions, six additional instructions are available with the 6301. Four of the instructions use direct (base page) or indexed (X register) addressing, perform logical operations on immediate data, and place the result in memory. These instructions are AIM, OIM, EIM, and TIM. The remaining two instructions XGDX and SLP, are special instructions, which will be explained in the next section.

In the following sections, the instructions, operand rules and conventions, and types of errors are explained.

INSTRUCTIONS

In this section, there is a description of each instruction, followed by a table with the mnemonic symbol, object code, addressing mode, Boolean operation, and the condition code register. The following symbols are used in the descriptions.

*	Boolean AND
+	Arithmetic addition
+	Exclusive OR
-->	Transfer
<-->	Exchange
R	Bit = 0
n	Not affected by operation
a	Affected by operation
Dir	Direct addressing mode
Ind	Indexed addressing mode
#	Immediate Data
H	Half-carry from bit 3
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7

Descriptions

And Immediate (AIM)– Evaluates the AND of the immediate data and the contents of memory, and places the result in memory.

Or Immediate (OIM)– Evaluates the OR of the immediate data and the contents of memory, and places the result in memory.

Exclusive Or Immediate (EIM)– Evaluates the EOR of the immediate data and the contents of memory, and places the result in memory.

Test Immediate (TIM)– Evaluates the AND of the immediate data and the contents of memory, and changes the flag of the associated condition code register.

Each of these instructions has three bytes. The first byte is the opcode, the second byte is immediate data, and the third byte is an address or offset and the address modifier.

The two remaining instructions are special instructions. The XGDX instruction exchanges the contents of the accumulator and the index register: (ACCD) <--> (IX). The SLP instruction puts the 6301 in a sleep mode with the contents of the registers secure and the peripherals of the MPU operational.

Table B-1. 6301 Instruction Summary

MNEMONIC	OBJECT CODE	ADDR MODE	OPERATION	CONDITION CODES					
				H	I	N	Z	V	C
				5	4	3	2	1	0
AIM	71	Dir	(M) * (IMM) --> (M)	n	n	a	a	R	n
	61	Inx							
OIM	72	Dir	(M) + (IMM) --> (M)	n	n	a	a	R	n
	62	Inx							
EIM	75	Dir	(M) + (IMM) --> (M)	n	n	a	a	R	n
	65	Inx							
TIM	7B	Dir	(M) * (IMM)	n	n	a	a	R	n
	6B	Inx							

Operand Rules and Conventions

The AIM, OIM, EIM, and TIM instructions must use the following format.

BYTE 1	BYTE 2	BYTE 3
Opcode	Operand1	Operand2[,Address modifier]

There must be one or more spaces between each byte.

Operand1. The first operand must be immediate data; therefore, the assembler will allow values and labels with or without the immediate data symbol (#). A list of acceptable formats and examples follows.

FORMAT	EXAMPLE
Value	10H
Label	VAL1
Immediate Value	#10H
Immediate Label	#VAL1
Expression Excluded Label	10H+0AH
Expression Included Label	VAL1+0AH, VAL1+VAL2
Immediate Expression Excluded Label	#10H+0AH
Immediate Expression Included Label	#VAL1+10H, #VAL1+VAL2
Label Declared As External or Expression Included Label Declared As External	EXT VAL1 EIM #VAL1 10H,X EIM #VAL1+10H 10H,X

NOTE

The error free legal range for values and values of labels is -7FH to 0FFH.

Operand2. The second operand is an 8-bit address in the direct addressing mode or an offset value to the index register in the index addressing mode. A list of acceptable formats and examples follows.

FORMAT	EXAMPLE
Value	10H
Label	0SFT1
Expression Excluded Label	10H+0AH
Expression Included Label	0FST1+0AH, 0FST1+0FST2
Label Declared As External or Expression Included Label Declared As External	EXT 0FST1 EIM 10H 0FST1,X EIM 10H 0FST1+10H,X

NOTE

The error free legal range for values and values of labels is 00H to 0FFH.

Address Modifier. The address modifier is optional; it defines the addressing mode (either direct or indexed for AIM, OIM, EIM, and TIM). If there is no address modifier, the default is the direct addressing mode. In indexed addressing, if the offset to the index register is zero, the second operand can be omitted. The formats and an example of each follows.

DIRECT FORMAT

EXAMPLE

<value or label or expression>,D

OIM 10H 10,D

INDEXED FORMAT

EXAMPLES

<value or label or expression>,X

OIM 10H ,X

Offset to index register is 0

OIM 10H ,X

OIM 10H X

TYPES OF ERRORS

The three types of errors are described in this section. Each type is explained and then an example is given.

Invalid Operand Error

The assembler will generate this error for any illegal format in the statement. If the second operand is one of the register symbols (A, B, or D) or one of the address modifiers D or E (X is allowed), this error will be generated. When this error occurs, no object code is generated (no located space is generated).

Example:

OIM

10H A

Legal Range Error

If an operand is out of the legal range, the assembler will generate this error. The object code generated will assume the operand value is zero.

Example:

```
      0IM      131H      10H
```

Expression Type Error

If the first operand is one of the register symbols A, B, or D, or any of the address modifiers D, E, or X, the assembler will generate this error. If the first and/or second operand has more than two labels declared as external, this error will be generated. The object code generated will assume the operand value is zero.

Example:

```
      0IM      A      10H,X
```